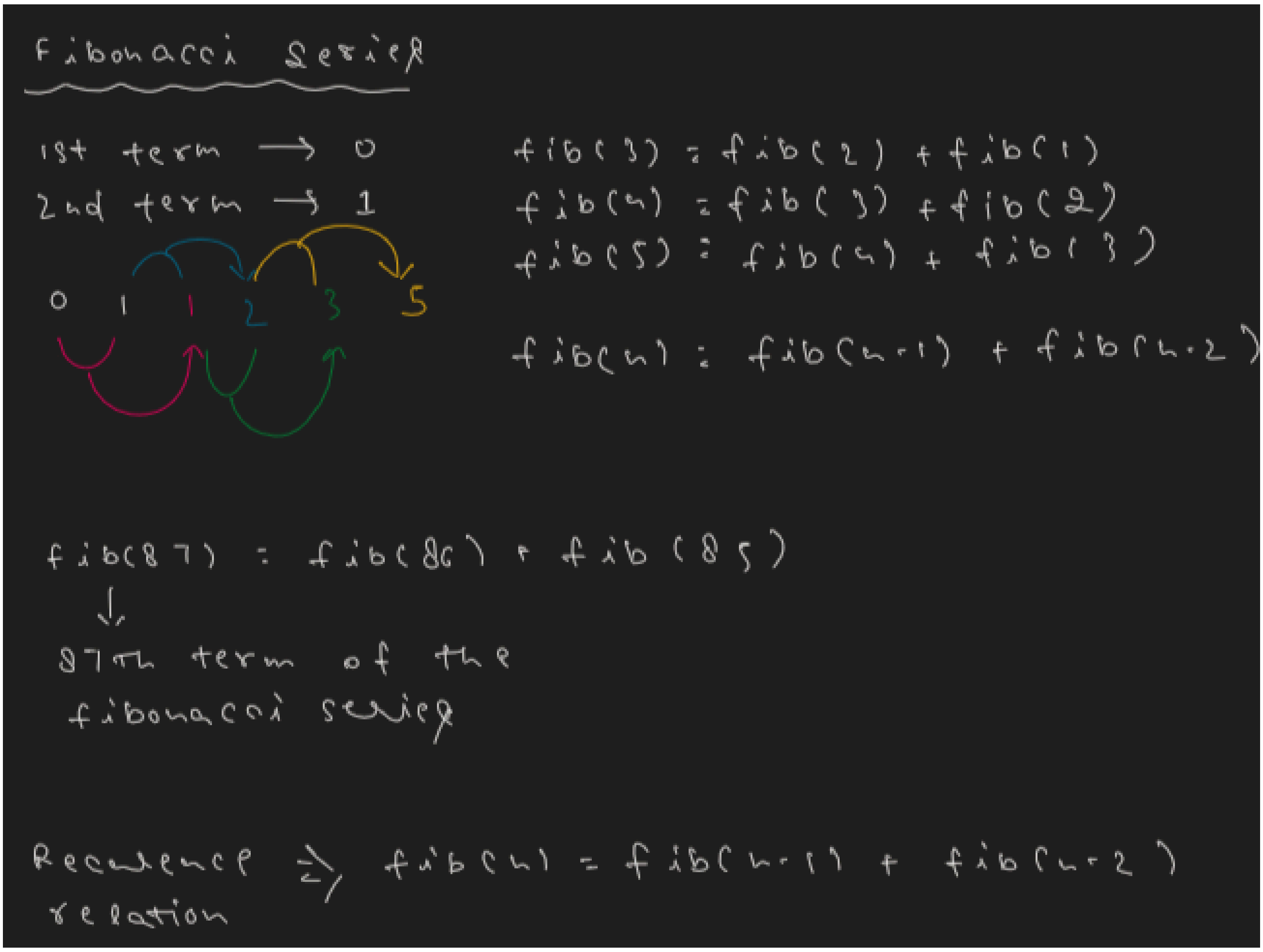
🎯 Fibonnaci nth term

https://leetcode.com/problems/fibonacci-number/



Fibonacci Series

1st term → 0

2nd term → 1

fib(3) = fib(2) + fib(1)

fib(4) = fib(3) + fib(2)

fib(5) = fib(4) + fib(3)

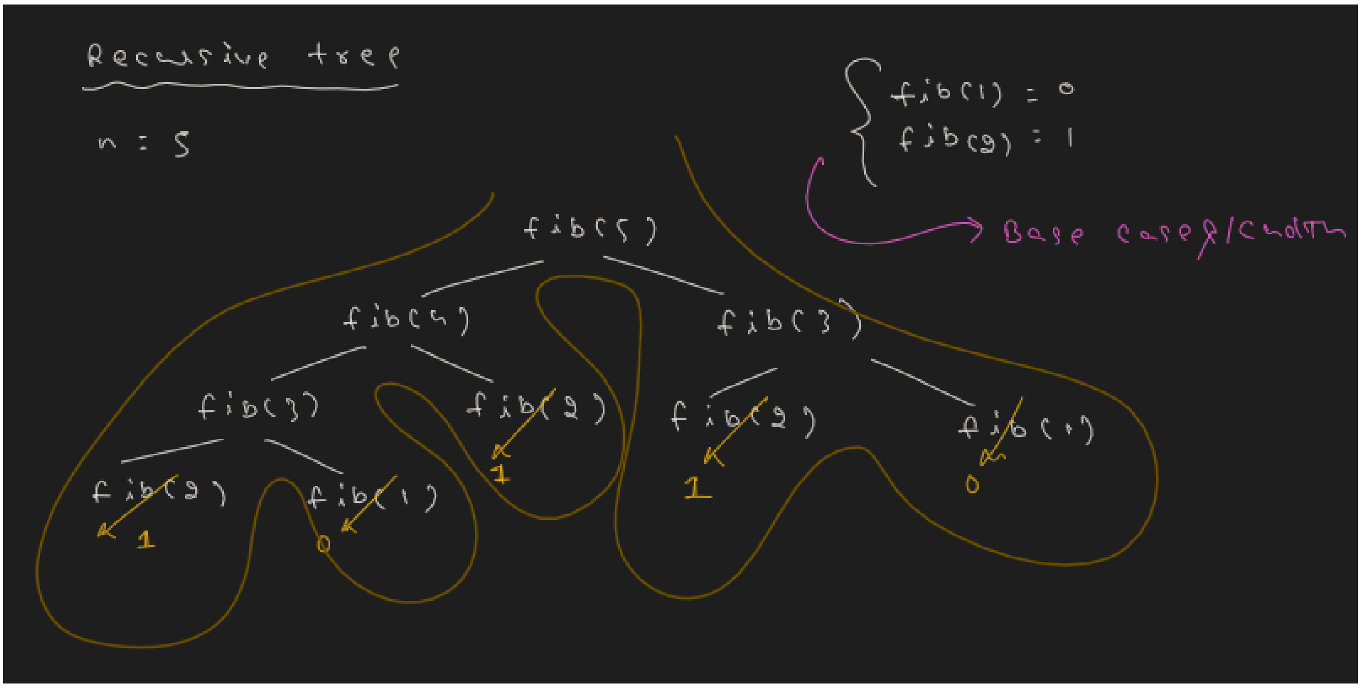
fib(n) = fib(n-1) + fib(n-2)

fib(87) = fib(86) + fib(85)

↓

87th term of the fibonacci series

Recurrence relation ⇒ fib(n) = fib(n-1) + fib(n-2)



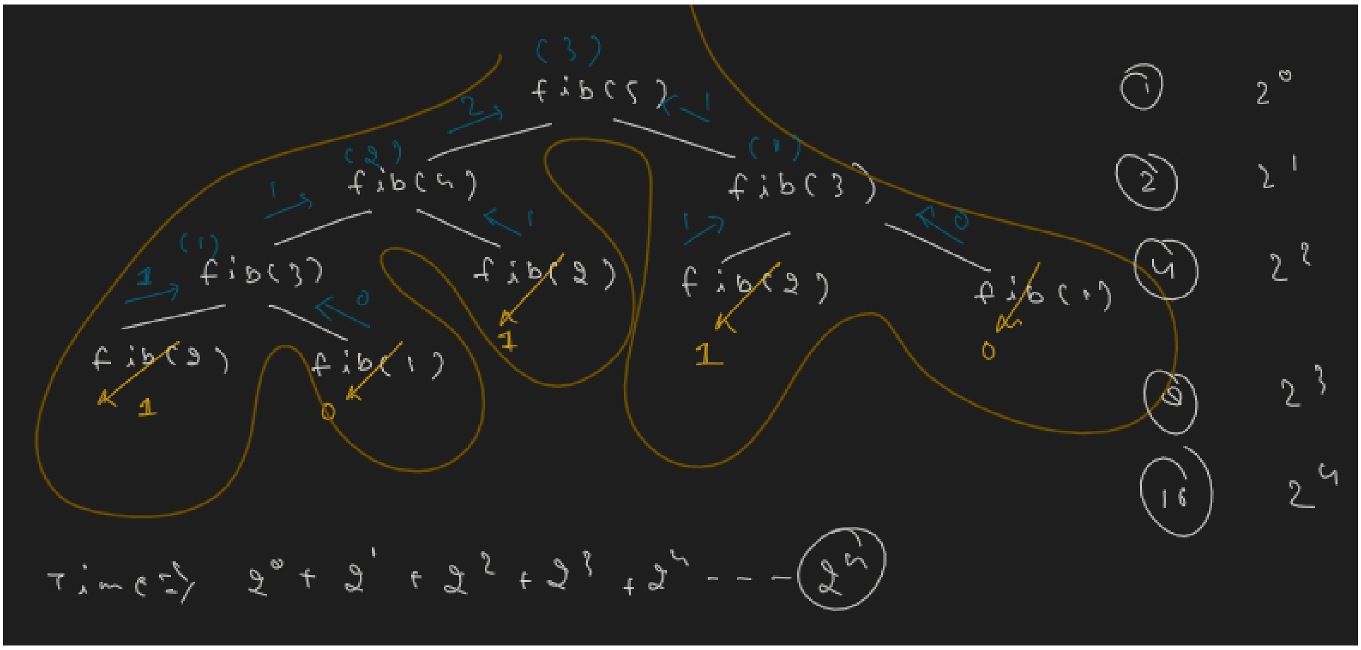
Recursive tree

n = 5

fib(1) = 0

fib(2) = 1

→ Base cases/Condn

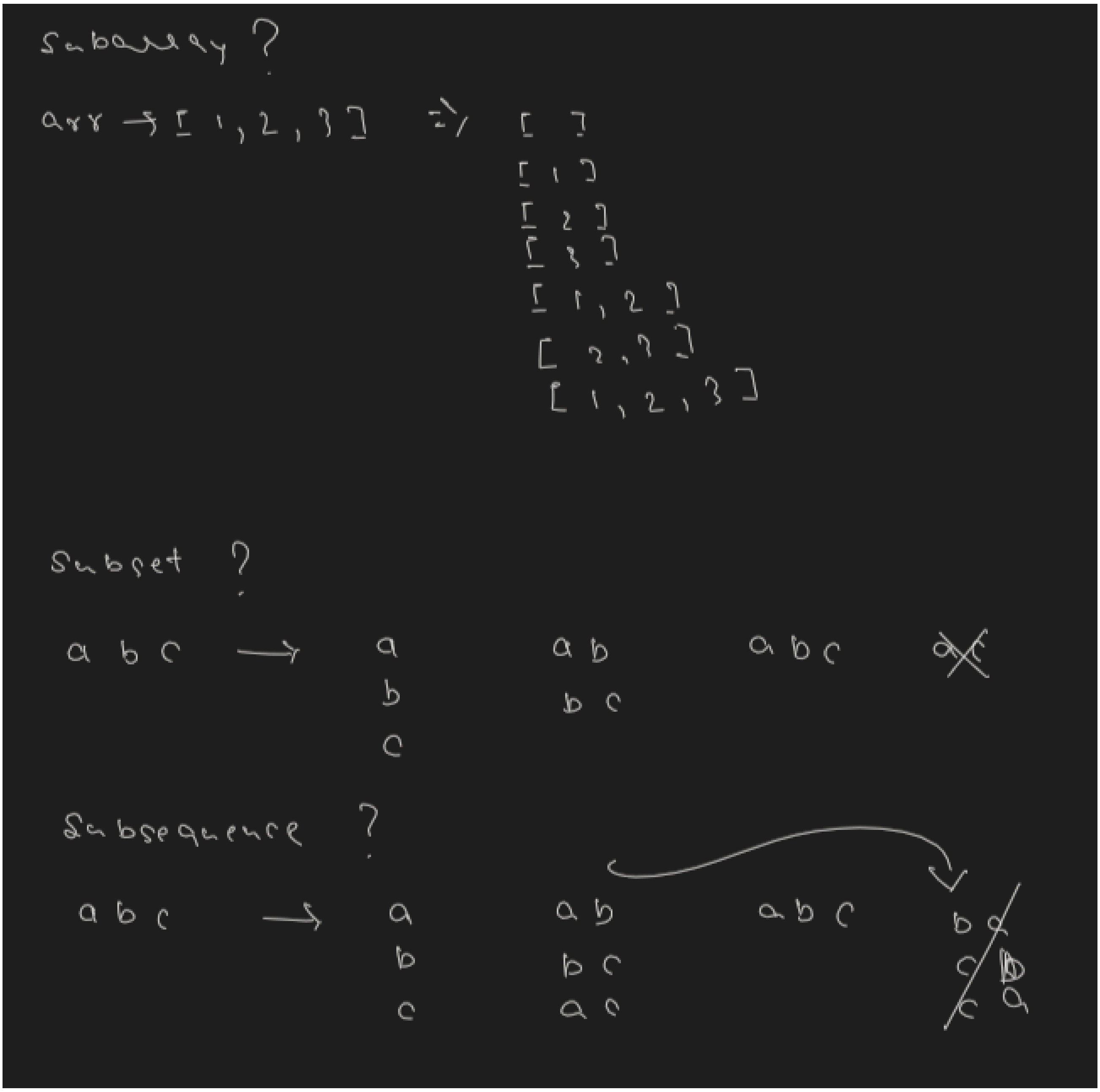


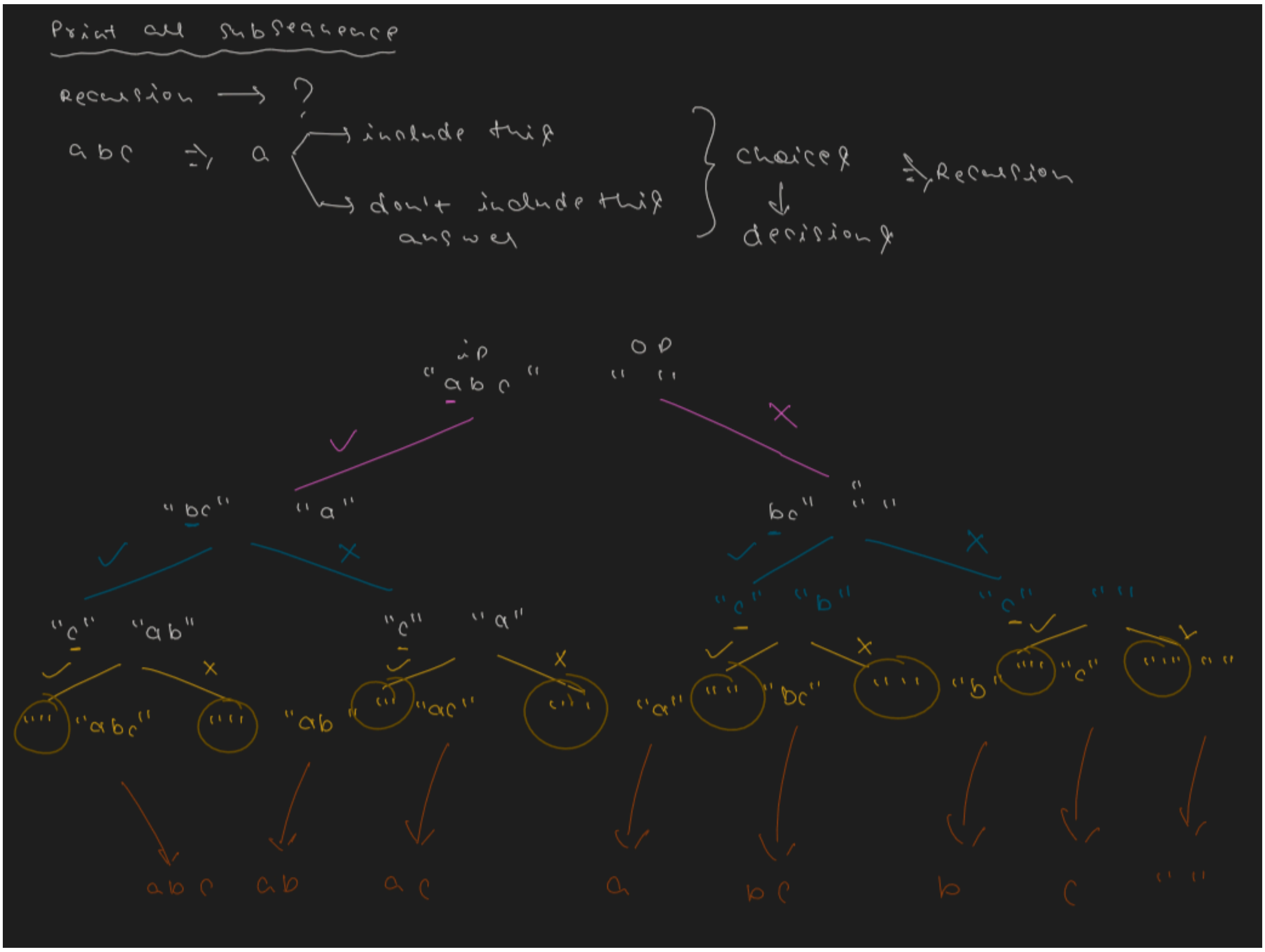
```
public class Main {
      public static void main(String[] args) {
            int n = 5;
            System.out.println(fib(n));
      }

      public static int fib(int n) {
            if (n == 0 || n == 1) {
                  return n;
            }
            return fib(n - 1) + fib(n - 2);
      }
}
```

```
class Solution {
    public int fib(int n) {
        if (n == 0 || n == 1) {
            return n;
        }
        return fib(n - 1) + fib(n - 2);
    }
}
```

Time → O(2^n)

🎯 Print all subsequence

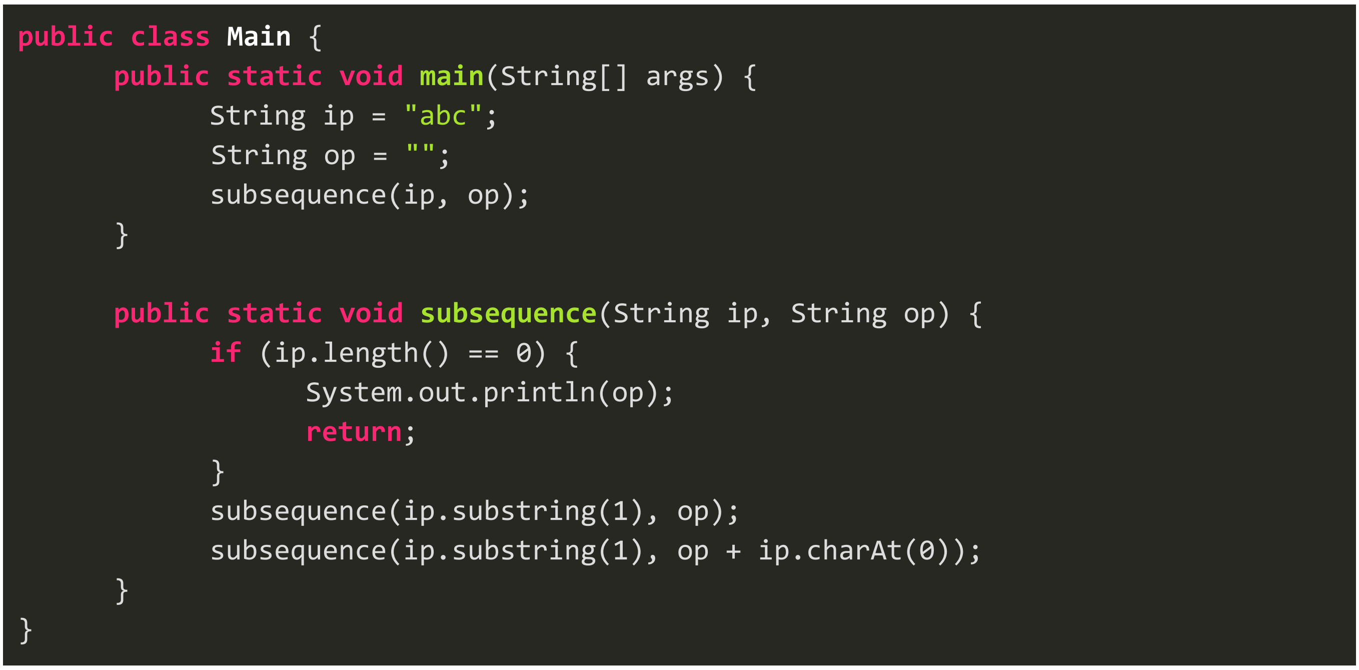




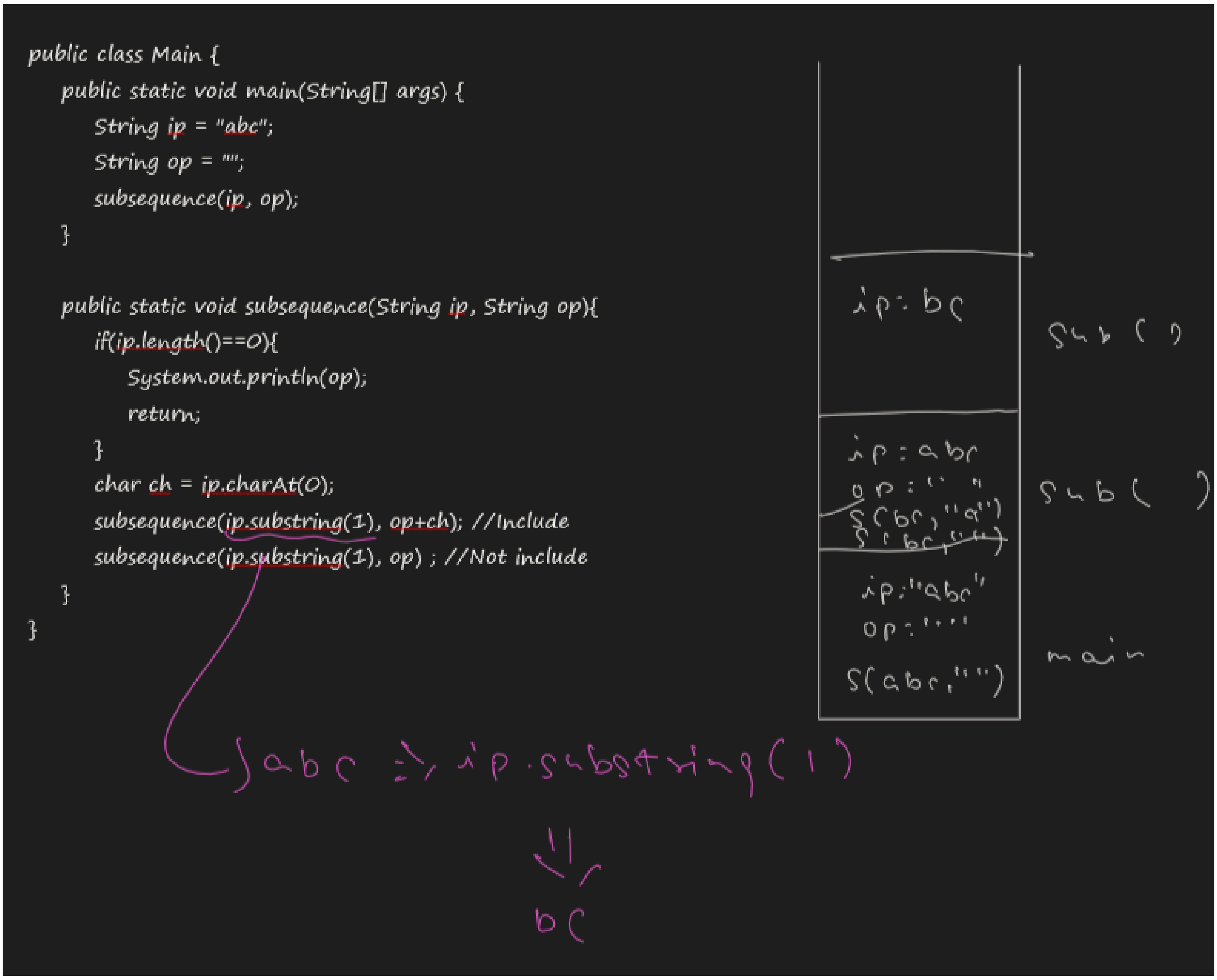
Print all subsequence

Recursion ⟶ ?

abc ⇒ a ⟶ include this / don't include this answer } choices ↓ decisions ⇒ Recursion



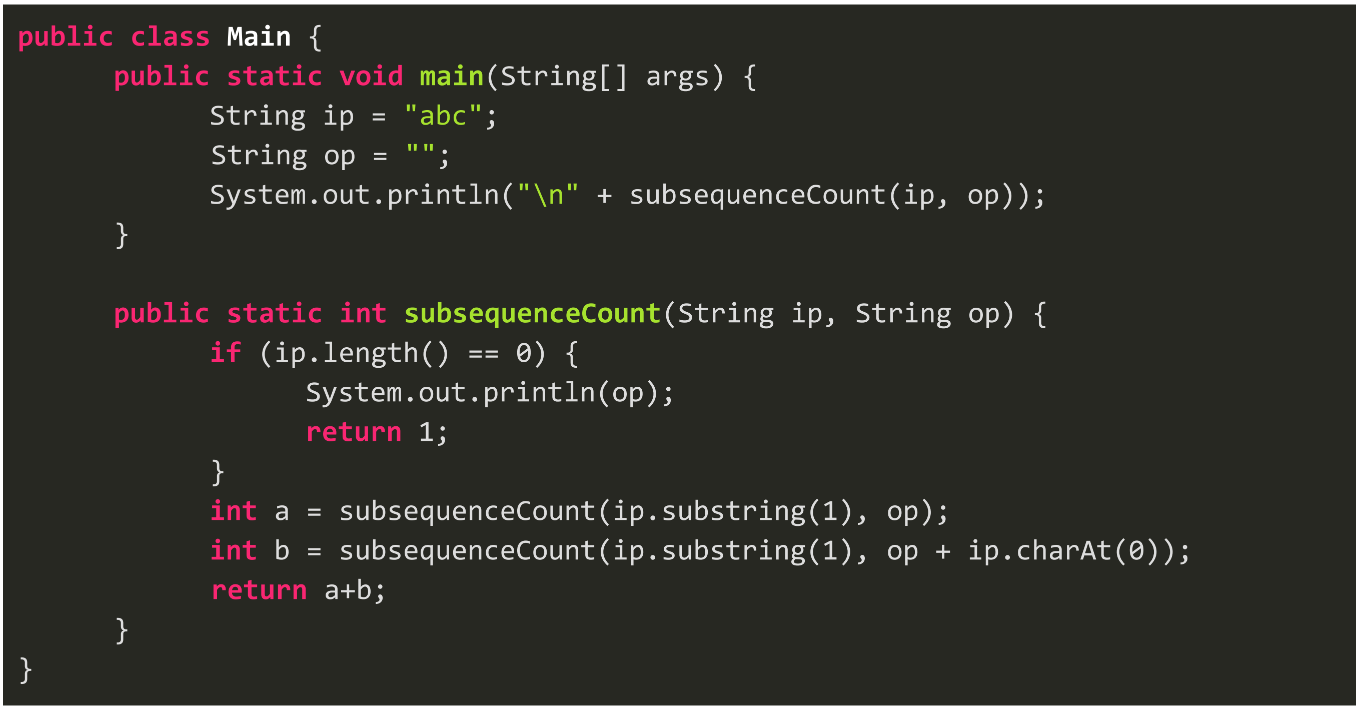
```java
public class Main {
	public static void main(String[] args) {
		String ip = "abc";
		String op = "";
		subsequence(ip, op);
	}

	public static void subsequence(String ip, String op) {
		if (ip.length() == 0) {
			System.out.println(op);
			return;
		}
		subsequence(ip.substring(1), op);
		subsequence(ip.substring(1), op + ip.charAt(0));
	}
}
```



```
public class Main {
    public static void main(String[] args) {
        String ip = "abc";
        String op = "";
        subsequence(ip, op);
    }

    public static void subsequence(String ip, String op){
        if(ip.length()==0){
            System.out.println(op);
            return;
        }
        char ch = ip.charAt(0);
        subsequence(ip.substring(1), op+ch); //Include
        subsequence(ip.substring(1), op) ; //Not include
    }
}
```

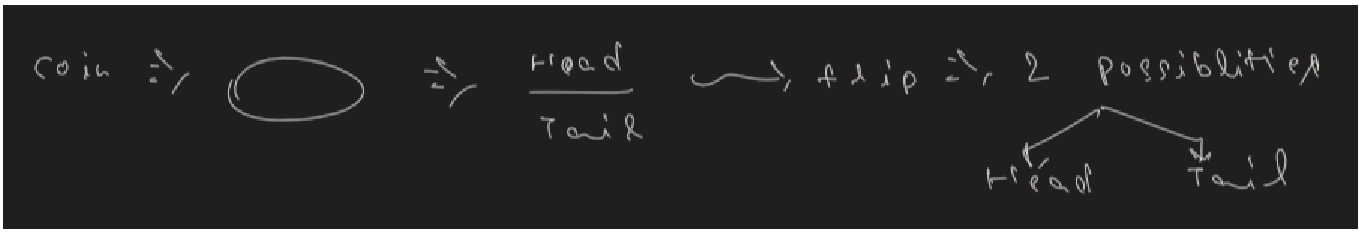
🎯 Print all subsequence count

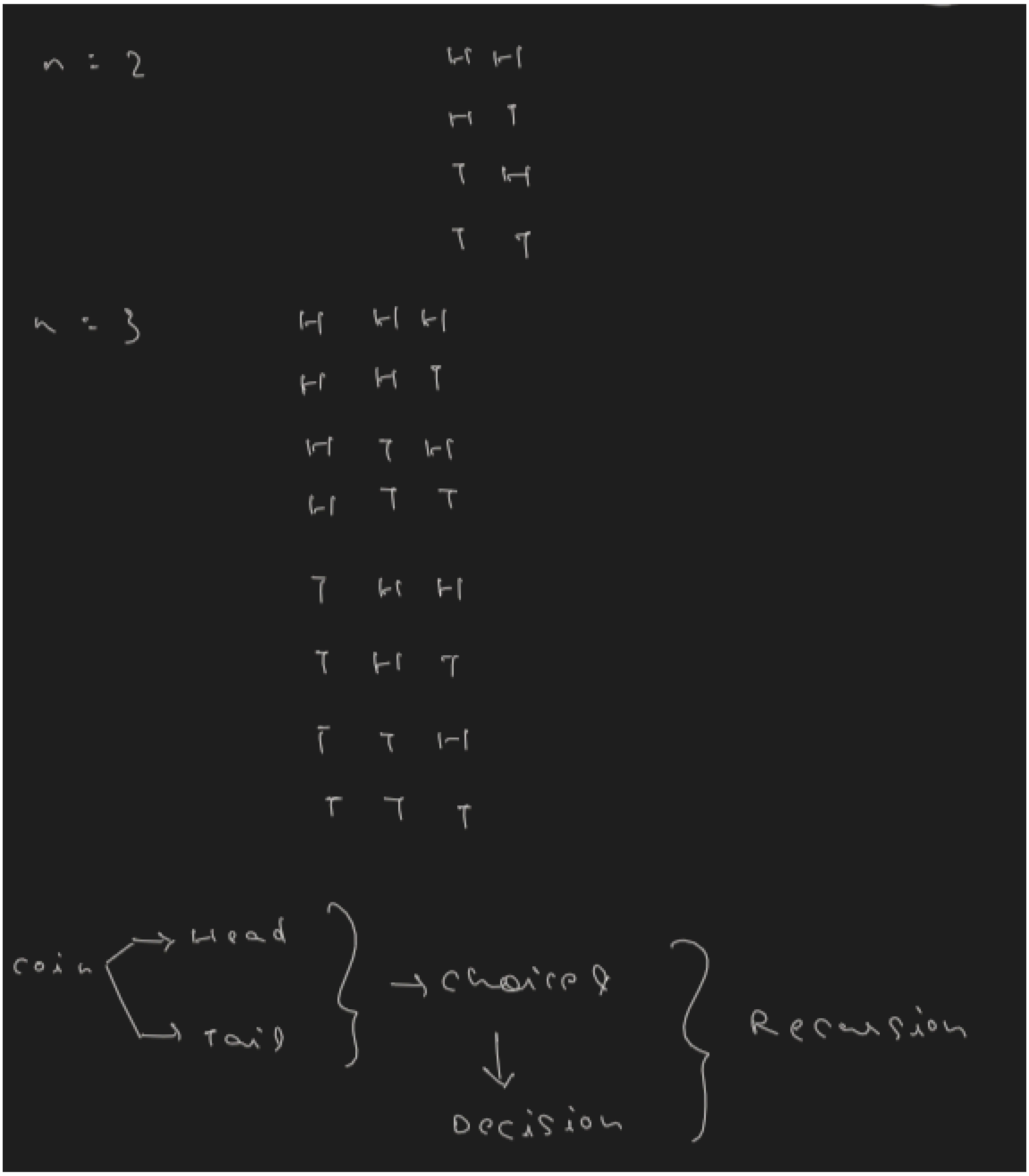
```
public class Main {
      public static void main(String[] args) {
            String ip = "abc";
            String op = "";
            subsequenceCount(ip, op);
            System.out.println("\n" + count);
      }

    static int count = 0;
      public static void subsequenceCount(String ip, String op) {
            if (ip.length() == 0) {
                  System.out.println(op);
            count++;
                  return;
            }
            subsequenceCount(ip.substring(1), op);
            subsequenceCount(ip.substring(1), op + ip.charAt(0));
      }
}
```



```java
public class Main {
      public static void main(String[] args) {
            String ip = "abc";
            String op = "";
            System.out.println("\n" + subsequenceCount(ip, op));
      }

      public static int subsequenceCount(String ip, String op) {
            if (ip.length() == 0) {
                  System.out.println(op);
                  return 1;
            }
            int a = subsequenceCount(ip.substring(1), op);
            int b = subsequenceCount(ip.substring(1), op + ip.charAt(0));
            return a+b;
      }
}
```

🎯 Print all possible outcome of coin flip



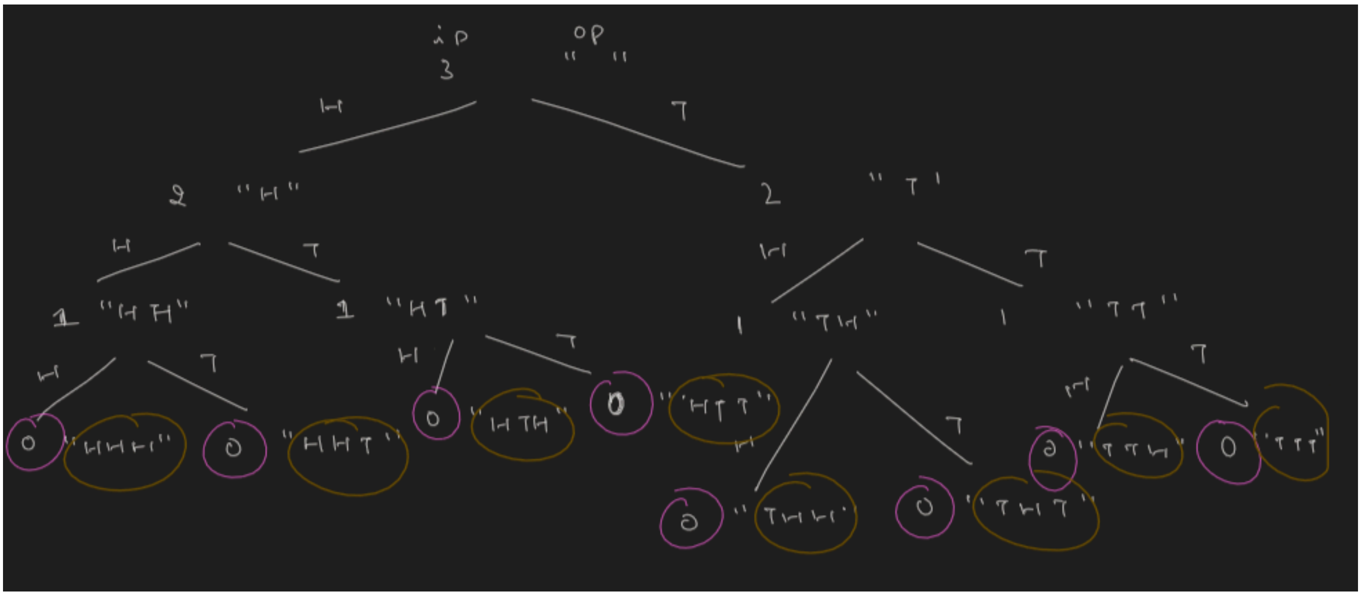


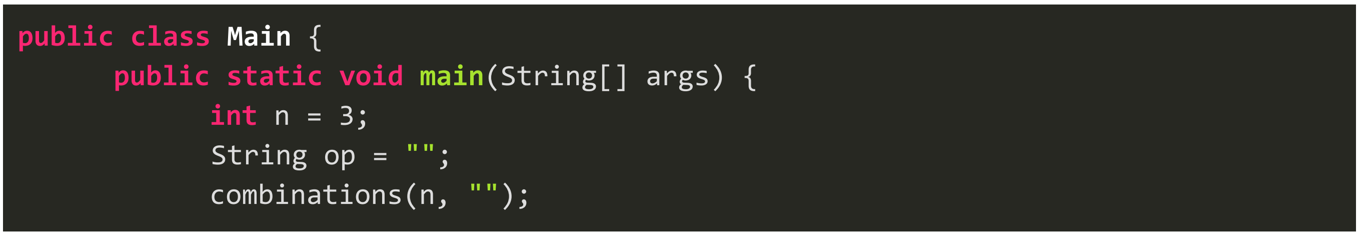
n = 2

H H
H T
T H
T T

n = 3

H H H
H H T
H T H
H T T
T H H
T H T
T T H
T T T





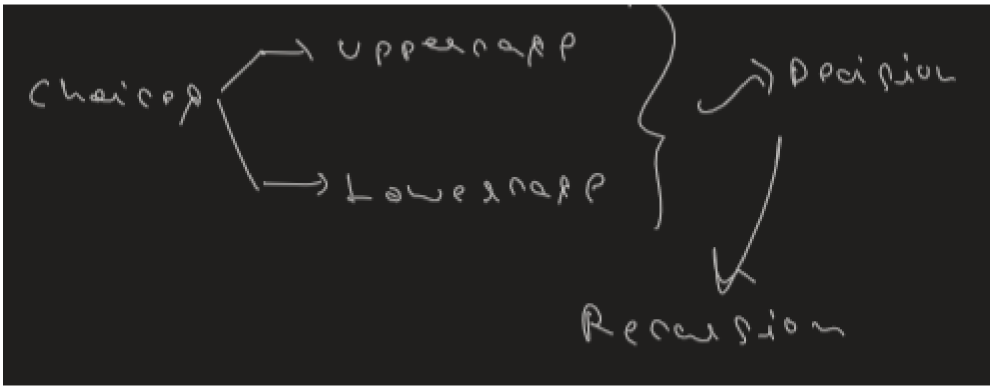
```
public class Main {
      public static void main(String[] args) {
            int n = 3;
            String op = "";
            combinations(n, "");
      }

      public static void combinations(int n, String op) {
            if (n == 0) {
                  System.out.println(op);
                  return;
            }
            combinations(n - 1, op + "H");
            combinations(n - 1, op + "T");
      }
}
```

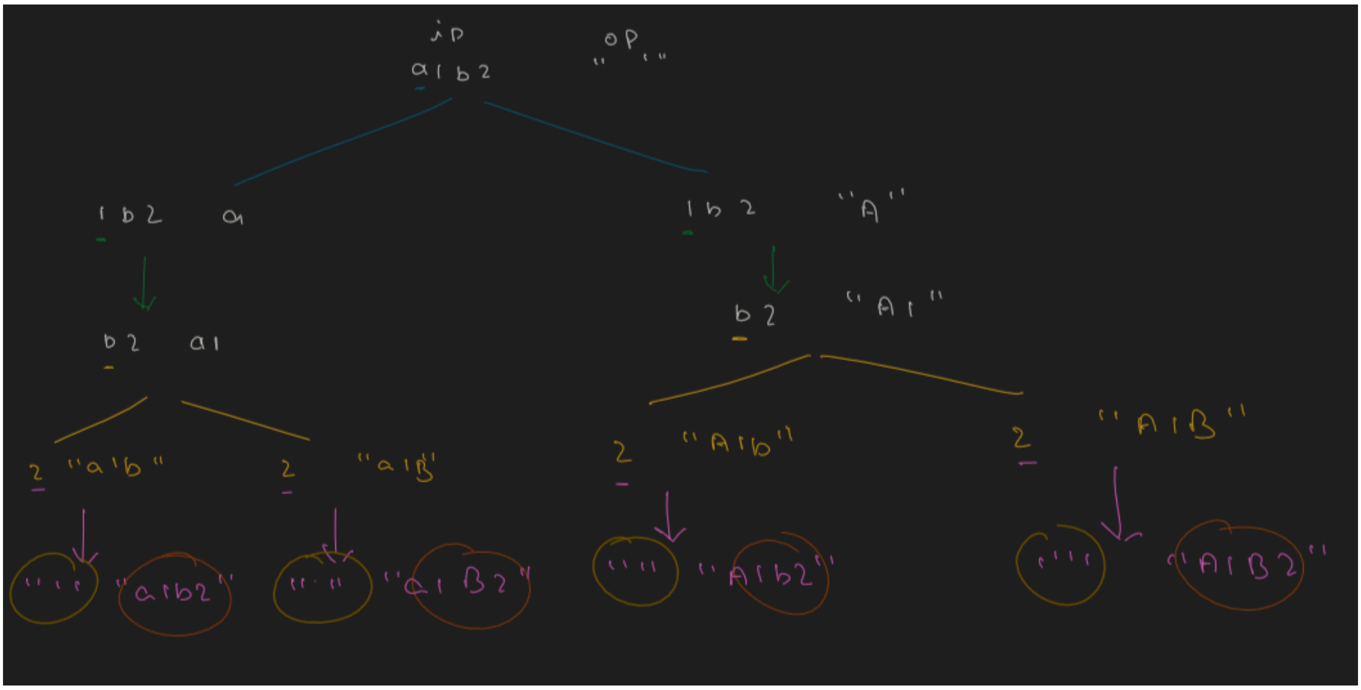
🎯 Print count of all possible outcome of coin flip

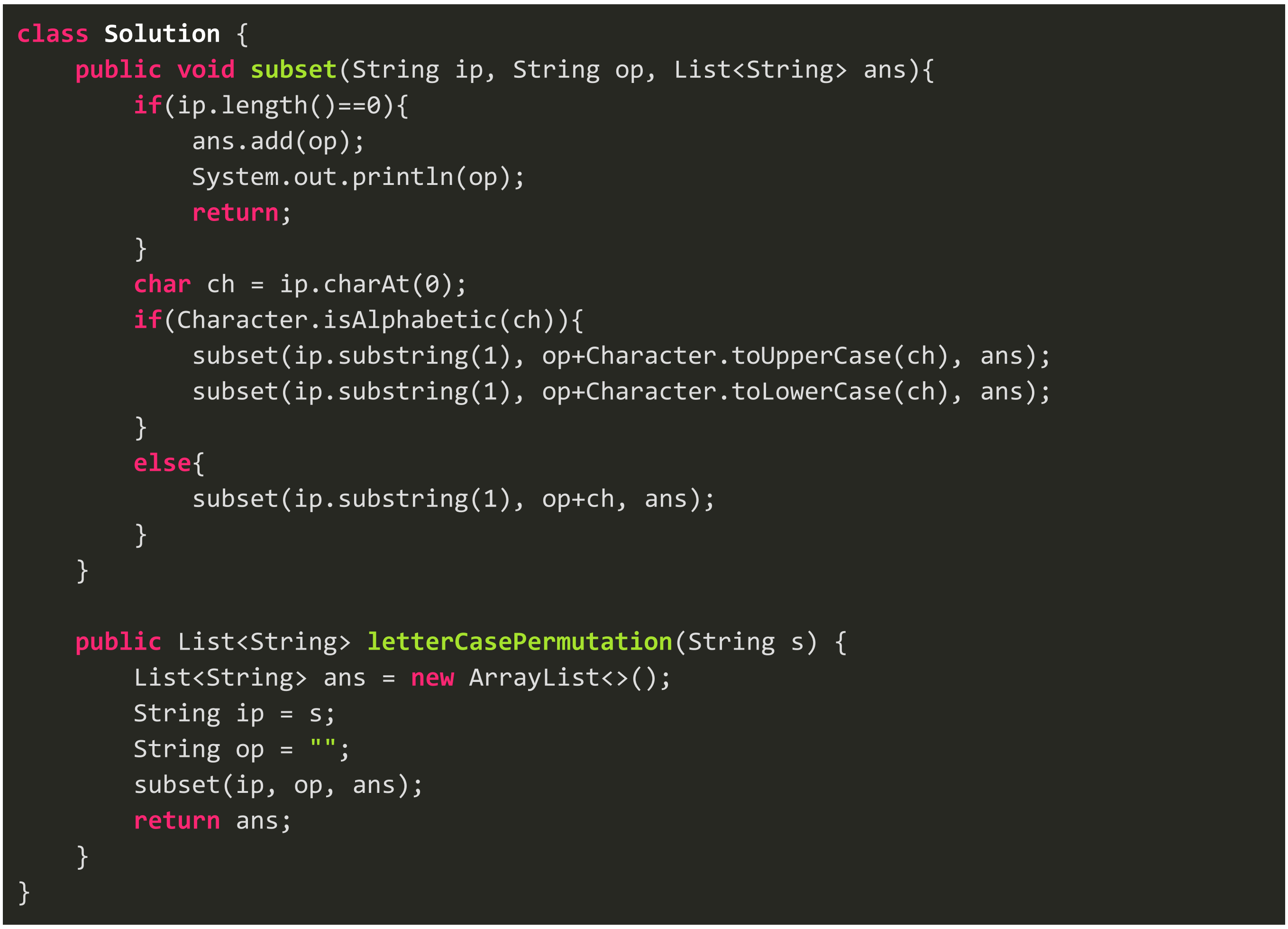
```
public class Main {
    public static void main(String[] args) {
            int n = 4;
        String op = "";
        System.out.println("\n" + combinations(n, ""));
      }

      public static int combinations(int n, String op) {
            if (n == 0) {
                  System.out.println(op);
                  return 1;
            }
            int a = combinations(n - 1, op + "H");
            int b = combinations(n - 1, op + "T");
        return a+b;
      }
}
```

🎯 Letter Case Permutation

https://leetcode.com/problems/letter-case-permutation/







```java
class Solution {
    public void subset(String ip, String op, List<String> ans){
        if(ip.length()==0){
            ans.add(op);
            System.out.println(op);
            return;
        }
        char ch = ip.charAt(0);
        if(Character.isAlphabetic(ch)){
            subset(ip.substring(1), op+Character.toUpperCase(ch), ans);
            subset(ip.substring(1), op+Character.toLowerCase(ch), ans);
        }
        else{
            subset(ip.substring(1), op+ch, ans);
        }
    }

    public List<String> letterCasePermutation(String s) {
        List<String> ans = new ArrayList<>();
        String ip = s;
        String op = "";
        subset(ip, op, ans);
        return ans;
    }
}
```